CROSSWORD (WORD SEARCH) PUZZLE GAME

MUHAMMAD ANAS HASSAN 20K-1726
MUHAMMAD WARZAN 20K-1649

INTERNAL ADVISOR DR. ASIM-UR-REHMAN

INTERNAL CO-ADVISOR DR. ASIM-UR-REHMAN

EXTERNAL ADVISOR DR. MUHAMMAD SHAHID SHEIKH

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES – FAST

JUNE 2010

ABSTRACT:

The purpose of this study was to see if playing a word search puzzle game influences learners' vocabulary growth. The participants in this study were junior high school students at a language school in Semnan. Researchers used the Standardized Comprehensive English Test to minimize a sample of 100 individuals to 60 homogenous pupils in order to meet the study's objectives. The participants were then divided into two groups of 30 students each at random. The first is an experiment, while the second is a control. Following that, both experimental and control individuals were given a pretest of 45 pre-validated multiple-choice vocabulary pieces in the first session. The experimental group got word search puzzles as treatment for the next eight sessions, while the control group received typical vocabulary exercises. Subjects in both groups took the same vocabulary exam as the post-test at the end of the semester to assess their progress.

INTRODUCTION:

Although Assembly language is not much commonly in use nowadays like other High Level programming languages but still it plays an important role in building hardware base implementation because it executes faster than high level.

Therefore, our aim and motivation are to design Crossword Game project in Assembly language. It is a type of word puzzle that consists of letters of words placed in a grid in which you have to find hidden letters of words from the grid in all directions i.e. up, down, right, left and also in diagonal. Word searches are commonplace in newspapers and magazines.

LITERATUTRE REVIEW:

This word search game is appropriate for kids 7 and above, and it appears to work nicely with three to four people. It took two players to complete the task, but it was uninteresting. The goal of the game is to locate as many words as possible. The hidden words on the themed cards (such as school, home, cities, and so on) may be found in all directions: horizontal, vertical, diagonal, left to right, and right to left. When you discover the word, you mark it with the colorful pieces that resemble contact lenses. When other players remove markers from the game, it becomes competitive.

PROBLEM DEFINITION:

Word searches are commonplace in newspapers and magazines. To solve this crossword puzzle problem, we divided the whole problem into sub problems called levels and each level difficulty level increases step by step. The problem in this crossword puzzle game defines that the user has to find the hidden letters of words in the grid, if the user inputted the right letters of word, then the score is incremented and if the user didn't input the right letters of word, then the score is not incremented whereas the live of the user are decremented. The object of the game is to find the most words. The hidden words on the themed cards (i.e., school, home, cities, etc.) are in all directions: horizontal, vertical, diagonal, from left to right, and right to left.

METHODOLOGY:

The method is that first of all, it reads the crossword puzzle game grid from the preferred level1.txt file and then prints it on the console and then ask the user to enter the letter of words from the hidden words in grid and then it compares. How it compares, we have three arrays for three levels each of size five and of byte type named as arr_L1, arr_L2 and arr_L3 and stores 1 in each of its elements. There are also three more arrays in the program named as word_list, word_list1, word_list2 each of byte type and of five size which store the correct/right letters of words from the grid, example five letters of words from level1 grid to word_list array and five letters of words from level2 grid to word_list1 array and also five letters of words from level3 grid to word_list2 array.

The whole process/working is that for level1 we move the value of arr_L1[0] to al register and then compare al and 1 such that if they are equal then it compares the two strings first one is the inputted string by the user and the next is the string element of word_list[0] using repe cmpsb. If these two strings are equal then it prints the statement that your entered word is equal and if these two strings are not equal then it prints that your entered word not found. And if user entered the right word then the score is also incremented and also the lives are decremented whenever the user enters the incorrect word. We also move zero to the arr_L1[0] because when user enters the second word in the level1 it agains comapres the value in arr_L1[0] with 1 and it is not equal next time and then it jumps to the next statement. This is the reason why we move zero to arr_L1 first element. And this whole process continues as it is for the rest values of the level and also for level2 and level3 only the array values and name of the array will be changed.

This Crossword Puzzle Game uses tools such as arrays, loops statements, jump conditional statements, procedures (functions), string functions, for formatting we have used settextcolor built in function to set the foreground and the background colors and different assembly language keywords.

The method/approach, technique and algorithms are sufficiently discussed with sufficient details.

DETAILED DESIGN AND ARCHITECTURE:

Our Crossword game consists of three levels and each level consists of crossword puzzle grid and are stored in a file and difficulty level is increased level by level. The user which plays this crossword puzzle game has total number of 5 lives. On every wrong input the lives are decreased by one and on every right input the score of the user is incremented by one. On the start of the game there are some choices for the user before playing the game i.e Quick play, Instruction, Setting, High score and the last is Quit game.

The Quick Play option in the start menu begins the crossword puzzle game with amazing three levels. The user which plays this crossword puzzle game has total number of 5 lives. On every wrong input the lives are decreased by one and on every right input the score of the user is incremented by one. In each level user have to enter five right letters of words in order to pass that level and to move towards the next level and same instructions as for the third level. If the lives are ended means zero then it displays the message that no lives let to play, GAME OVER. And if user completed all the three levels then the score of the user is 15/15.

The Instruction option in the start menu contains the instruction about the game before starting the game. All the instructions are stored in a txt file so, it reads the file and then print it in the instructions option. It is important for the user to follow the instructions when the user is playing the game so, the user should read the instructions before starting the game.

The setting option in the start menu contains 3 more nested formatting operations one is the change font color, second is the change background color and the third one is change both font and background color. In change font color some font colors are preferred to the user. User has to select one of them to set the desired font color of the crossword puzzle game. The default background color at that time is black. In change background color some background colors are preferred to the user. User has to select one of them to set the desired background color of the crossword puzzle game. The default font color at that time is black. In change both font and background color some combinations of both font and background colors are preferred to the user. User has to select one of those combinations to set the font color and the background color of the crossword puzzle game.

The Quit option in the start menu quit the program if the user doesn't wants to play the game.

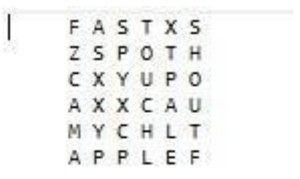
If we talk about Architecture, we have used Microsoft Macro Assembler (masm) architecture with two libraries i.e Irvine32.inc and Macros.inc and the IDE is Visual Studio 2019.

IMPLEMENTATION TESING AND PROGRAMMING CODE:

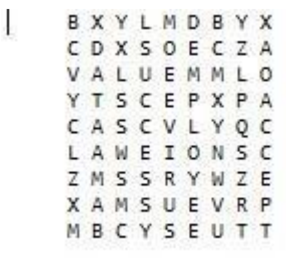
This Crossword Puzzle Game uses tools such as arrays, loops statements, jump conditional statements, procedures (functions), strings functions, for formatting we have used settextcolor built in function to set the foreground and the background colors and different assembly language keywords.
All the functions have no parameters/ arguments and also have no return type.
For all these implementation we have made eleven functions in our program i.e

- Quick_play PROC
- Instruction PROC
- Setting PROC
- ChangeFontColor PROC
- ChangeBackgroundColor PROC
- ChangeBothFontAndBackgroundColor PROC
- Level1 PROC
- Level2 PROC
- Level3 PROC
- Read_File PROC
- Write_File PROC

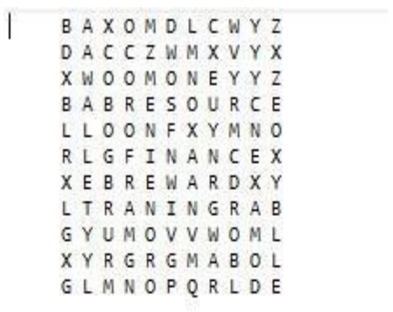
Also we created 4 files i.e level1, level2, level3 which consists of different grids and instruction which consists of instructions before playing the game.



```
F A S T X S
Z S P O T H
C X Y U P O
A X X C A U
M Y C H L T
A P P L E F
```



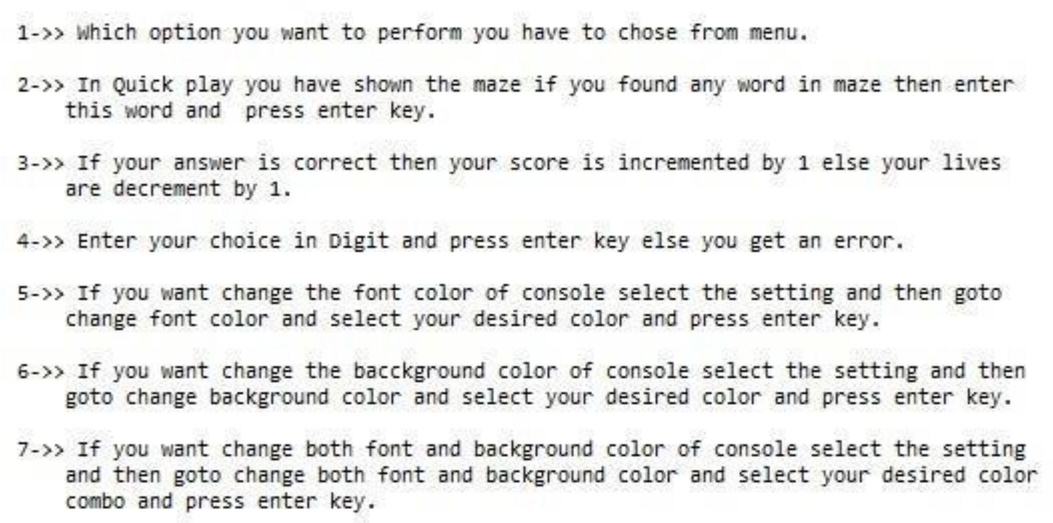
```
B X Y L M D B Y X
C D X S O E C Z A
V A L U E M M L O
Y T S C E P X P A
C A S C V L Y Q C
L A W E I O N S C
Z M S S R Y W Z E
X A M S U E V R P
M B C Y S E U T T
```



```
B A X O M D L C W Y Z
D A C C Z W M X V Y X
X W O O M O N E Y Y Z
B A B R E S O U R C E
L L O O N F X Y M N O
R L G F I N A N C E X
X E B R E W A R D X Y
L T R A N I N G R A B
G Y U M O V V W O M L
X Y R G R G M A B O L
G L M N O P Q R L D E
```

level1.txt

level2.txt level3.txt


```
1->> Which option you want to perform you have to chose from menu.

2->> In Quick play you have shown the maze if you found any word in maze then enter
    this word and  press enter key.

3->> If your answer is correct then your score is incremented by 1 else your lives
    are decrement by 1.

4->> Enter your choice in Digit and press enter key else you get an error.

5->> If you want change the font color of console select the setting and then goto
    change font color and select your desired color and press enter key.

6->> If you want change the bacckground color of console select the setting and then
    goto change background color and select your desired color and press enter key.

7->> If you want change both font and background color of console select the setting
    and then goto change both font and background color and select your desired color
    combo and press enter key.
```

instructions.txt

This is the code of Crossword Puzzle Game in Assembly Language:

```
Include Irvine32.inc
Include macros.inc
BUFFER_SIZE = 1000

.data
str1 BYTE "  Enter the Word : ",0
input BYTE 10 DUP(?)
score BYTE 0
Lives BYTE 5
check BYTE 1
word_list BYTE "FAST","APPLE","SPOT","TOUCH","SHOUT",0
word_list1 BYTE "VALUE","EMPLOYEE","SUCCESS","LAW","VIRUS",0
word_list2 BYTE "FINANCE","MONEY","REWARD","WALLET","WARE",0
arr_L1 BYTE 5 DUP(1)
arr_L2 BYTE 5 DUP(1)
arr_L3 BYTE 5 DUP(1)
file_L1 BYTE "level1.txt",0
file_L2 BYTE "level2.txt",0
file_L3 BYTE "level3.txt",0
file_L4 BYTE "instruction.txt",0
char BYTE 4 Dup("0")


;read file
buffer BYTE BUFFER_SIZE DUP(0)
fileHandle HANDLE ?
```

```
;write high score to file
filename BYTE "high_score.txt",0
stringLength DWORD ?

.code
main proc

Again:
     call clrscr
     call crlf
     mWrite<"     'CROSSWORD PUZZLE GAME PROJECT'",0dh,0ah>
  mWrite<"    ================================",0>
     call crlf
     call crlf
     call crlf
     mWrite<" 1- Quick Play",0dh,0ah," 2- Instruction",0dh,0ah," 3- Setting",0dh,0ah>
     mWrite<" 4- Quit",0dh,0dh,0ah,0ah>
     mWrite<"  Enter Choice : ",0>
     mov eax,0
     call readdec

     cmp al,1
     jne next
     call Quick_play
     jmp quit

     next:
         cmp al,2
         jne next1
            call clrscr
         call Instruction
         jmp quit

     next1:
          cmp al,3
          jne next2
          call Setting
          jmp quit

     next2:
          cmp al,4
          jne next3
          mov check,0
          jmp Quit1

     next3:
          mWrite <" You Enter Invalid Number",0dh,0ah>
          mov eax,1500
          call delay
```

```
            jmp Again

  quit:
           call readdec
           cmp check,0
           jne Again

Quit1:
      exit
main endp

;------------------*Quick_play*------------------
Quick_play PROC

call clrscr
call Level1

call clrscr
cmp lives,0
je gameover
call Level2

call clrscr
cmp lives,0
je gameover
call Level3
jmp quit

gameover:
         call crlf
             call crlf
             call crlf
             call crlf
         mWrite<"      NO LIFE LEFT TO PLAY",0>
             call Crlf
     call Crlf
         mWrite<"       <------ * GAME OVER!!!!! * ------>",0>
         jmp quit

quit:
     ret
Quick_play endp
;--------------------------------------------------

;------------------*Setting*------------------
Setting PROC
call clrscr
mWrite<" 1- Change Font Color ",0dh,0ah>
mWrite<" 2- Change Background Color ",0dh,0ah>
```

```
mWrite<" 3- Change Both Font and Background Color ",0dh,0ah>
call crlf
Again:
      mWrite<"  Enter Choice : ",0>
      mov eax,0
      call readdec
      cmp al,1
      jne next
      call ChangeFontColor
      jmp next2
  next:
       cmp al,2
               jne next1
               call ChangeBackgroundColor
               jmp next3
  next1:
       cmp al,3
               jne next2
               call ChangeFontAndBackgroundColor
               jmp next3
      next2:
           mWrite<"  You enter Invalid number",0dh,0ah>
           mov eax,1500
           call delay
           jmp Again

next3:
      ret
Setting endp
;-------------------------------------------------

;------------------*ChangeFontColor*------------------
ChangeFontColor PROC

call clrscr
mWrite<" 1- Blue",0dh,0ah," 2- White",0dh,0ah," 3- Green",0dh,0ah>
mWrite<" 4- Red",0dh,0ah," 5- Magenta",0dh,0ah," 6- Yellow",0dh,0ah>
mWrite<" 7- Cyan",0dh,0ah," 8- Brown",0dh,0ah>
call Crlf
mWrite<"  Select your desired Font Color : ",0>
mov eax,0
call readdec

cmp al,1
jne next
mov eax,blue
call settextcolor
jmp quit
```

```
next:
     cmp al,2
     jne next1
     mov eax,white
     call settextcolor
     jmp quit

next1:
      cmp al,3
      jne next2
      mov eax,green
      call settextcolor
      jmp quit

next2:
      cmp al,4
      jne next3
      mov eax,red
      call settextcolor
      jmp quit

next3:
      cmp al,5
      jne next4
      mov eax,magenta
      call settextcolor
      jmp quit

next4:
      cmp al,6
      jne next5
      mov eax,yellow
      call settextcolor
      jmp quit

next5:
      cmp al,7
      jne next6
      mov eax,cyan
      call settextcolor
      jmp quit

next6:
      cmp al,8
      jne next7
      mov eax,brown
      call settextcolor
      jmp quit
```

```
next7:
      mWrite <"  You Enter Invalid Number",0dh,0ah>

quit:
     ret
ChangeFontColor endp
;---------------------------------------------------

;-----------------***ChangeBackgroundColor***-----------------
ChangeBackgroundColor PROC

call clrscr
mWrite<" 1- Blue",0dh,0ah," 2- White",0dh,0ah," 3- Green",0dh,0ah>
mWrite<" 4- Red",0dh,0ah," 5- Magenta",0dh,0ah," 6- Yellow",0dh,0ah>
mWrite<" 7- Cyan",0dh,0ah," 8- Brown",0dh,0ah>
call Crlf
mWrite<"  Select your desired Background Color : ",0>
mov eax,0
call readdec

cmp al,1
jne next
mov eax,(blue*16)
call settextcolor
call clrscr
jmp quit

next:
     cmp al,2
     jne next1
     mov eax,(white*16)
     call settextcolor
 call clrscr
     jmp quit

next1:
      cmp al,3
      jne next2
      mov eax,(green*16)
      call settextcolor
  call clrscr
      jmp quit

next2:
      cmp al,4
      jne next3
      mov eax,(red*16)
      call settextcolor
  call clrscr
```

```
      jmp quit

next3:
      cmp al,5
      jne next4
      mov eax,(magenta*16)
      call settextcolor
  call clrscr
      jmp quit

next4:
      cmp al,6
      jne next5
      mov eax,(yellow*16)
      call settextcolor
  call clrscr
      jmp quit

next5:
      cmp al,7
      jne next6
      mov eax,(cyan*16)
      call settextcolor
  call clrscr
      jmp quit

next6:
      cmp al,8
      jne next7
      mov eax,(brown*16)
      call settextcolor
  call clrscr
      jmp quit

next7:
      mWrite <"  You Enter Invalid Number",0dh,0ah>

quit:
     ret
ChangeBackgroundColor ENDP
;-------------------------------------------------------------

;------------------***ChangeFontAndBackgroundColor***------------------
ChangeFontAndBackgroundColor PROC

call clrscr
mWrite<" 1- Blue(font) and Gray(background)",0dh,0ah," 2- White(font) and Red(background)",0dh,0ah,"
3- Green(font) and Black(background)",0dh,0ah>
```

```
mWrite<" 4- Red(font) and LightGreen(background)",0dh,0ah," 5- Magenta(font) and
LightRed(background)",0dh,0ah," 6- Yellow(font) and Magenta(background)",0dh,0ah>
mWrite<" 7- Cyan(font) and LightGray(background)",0dh,0ah," 8- Brown(font) and
LightBlue(background)",0dh,0ah," 9- Black(font) and White(background)",0dh,0ah>
call Crlf
mWrite<"  Select your desired Font and Background Color Combination: ",0>
mov eax,0
call readdec

cmp al,1
jne next
mov eax,blue+(gray*16)
call settextcolor
call clrscr
jmp quit

next:
    cmp al,2
    jne next1
    mov eax,white+(red*16)
    call settextcolor
 call clrscr
    jmp quit

next1:
     cmp al,3
     jne next2
     mov eax,green+(black*16)
     call settextcolor
  call clrscr
     jmp quit

next2:
     cmp al,4
     jne next3
     mov eax,red+(lightgreen*16)
     call settextcolor
  call clrscr
     jmp quit

next3:
     cmp al,5
     jne next4
     mov eax,magenta+(lightred*16)
     call settextcolor
  call clrscr
     jmp quit

next4:
     cmp al,6
     jne next5
     mov eax,yellow+(magenta*16)
     call settextcolor
  call clrscr
     jmp quit

next5:
     cmp al,7
     jne next6
     mov eax,cyan+(lightgray*16)
     call settextcolor
  call clrscr
     jmp quit

next6:
     cmp al,8
     jne next7
     mov eax,brown+(lightblue*16)
     call settextcolor
  call clrscr
     jmp quit

next7:
     cmp al,9
     jne next8
     mov eax,black+(white*16)
     call settextcolor
  call clrscr
     jmp quit

next8:
     mWrite <"  You Enter Invalid Number",0dh,0ah>

quit:
    ret
ChangeFontAndBackgroundColor ENDP
;-------------------------------------------------------------------

;-----------------*Level1*-----------------
Level1 PROC

outter:
       call crlf
       call crlf
       call crlf
       mov ecx,1000
       call delay
       mWrite<"       > LEVEL 1: <",0ah,0>
```

```
   call crlf
   call crlf
   cmp lives,0
je nolife

mWrite<" >> Lives : ",0>
   movzx eax,lives
   call writedec
   mWrite<"    >> Score : ",0>
   movzx  eax,score
   call WriteDec
call crlf
call crlf

mov edx,offset file_L1
call Read_File
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
   mov  ecx,9
   call ReadString

mov al,arr_L1[0]
cmp al,1
jne else1

cld
mov esi,offset input
mov edi,offset word_list[0]
mov ecx,4
repe cmpsb
jnZ else1
call crlf
mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
inc score
mov arr_L1[0],0
jmp next

else1:
     mov al,arr_L1[1]
     cmp al,1
     jne else2
     cld
     mov esi,offset input
     mov edi,offset word_list[4]
     mov ecx,5
```

```
       repe cmpsb
       jnz else2
                call crlf
       mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
       inc score
       mov arr_L1[1],0
       jmp next

else2:
        mov al,arr_L1[2]
      cmp al,1
      jne else3
      cld
      mov esi,offset input
      mov edi,offset word_list[9]
      mov ecx,4
      repe cmpsb
      jnz else3
                call crlf
      mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
      inc score
      mov arr_L1[2],0
      jmp next

    else3:
        mov al,arr_L1[3]
      cmp al,1
      jne else4
      cld
      mov esi,offset input
      mov edi,offset word_list[13]
      mov ecx,5
      repe cmpsb
      jnz else4
                call crlf
      mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
      inc score
      mov arr_L1[3],0
      jmp next

    else4:
        mov al,arr_L1[4]
      cmp al,1
      jne else5
      cld
      mov esi,offset input
      mov edi,offset word_list[18]
      mov ecx,5
      repe cmpsb
```

```
        jnz else5
                    call crlf
        mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
        inc score
        mov arr_L1[4],0
        jmp next

      else5:
        call crlf
            mWrite<" Your entered word not found!!!!",0dh,0ah>
        dec lives
      next:
           mov eax,1500
           call delay
           call clrscr
           mov al,score
           cmp al,5
           jl outter
           JZ quit

nolife:
     call crlf
 call crlf
 call crlf
 call crlf
 call crlf
     mWrite<"      NO LIFE LEFT TO PLAY",0>
     mWrite<"       <------ * GAME OVER!!!!! * ------>",0>
 ret

quit:
     ret
Level1 endp
;-----------------------------------------------

;------------------*Level2*------------------
Level2 Proc

   call crlf
   call crlf
   call crlf
   mov ecx,1000
   call delay
   mWrite<" --->>> LEVEL 1 SUCCESSFULLY COMPLETED, NOW PASS NEXT LEVEL <<<----",0>
outter:
       call crlf
   call crlf
   call crlf
   mWrite<"       > LEVEL 2: <",0>
```

```
call crlf
call crlf
   cmp lives,0
je nolife

   mWrite<" >> Lives : ",0>
   movzx eax,lives
   call writedec
   mWrite<"     >> Score : ",0>
   movzx eax,score
   call WriteDec
call crlf
call crlf
mov edx,offset file_L2
call Read_File
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
   mov  ecx,9
   call ReadString


mov al,arr_L2[0]
cmp al,1
jne else1
cld
mov esi,offset input
mov edi,offset word_list1[0]
mov ecx,5
repe cmpsb
jnZ else1
call crlf
mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
inc score
mov arr_L2[0],0
jmp next

else1:
     mov al,arr_L2[1]
     cmp al,1
     jne else2
     cld
     mov esi,offset input
     mov edi,offset word_list1[5]
     mov ecx,8
     repe cmpsb
     jnz else2
               call crlf
     mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
     inc score
     mov arr_L2[1],0
     jmp next

else2:
       mov al,arr_L2[2]
     cmp al,1
     jne else3
     cld
     mov esi,offset input
     mov edi,offset word_list1[13]
     mov ecx,7
     repe cmpsb
     jnz else3
               call crlf
     mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
     inc score
     mov arr_L2[2],0
     jmp next

else3:
       mov al,arr_L2[3]
     cmp al,1
     jne else4
     cld
     mov esi,offset input
     mov edi,offset word_list1[20]
     mov ecx,3
     repe cmpsb
     jnz else4
               call crlf
     mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
     inc score
     mov arr_L2[3],0
     jmp next

else4:
       mov al,arr_L2[4]
     cmp al,1
     jne else5
     cld
     mov esi,offset input
     mov edi,offset word_list1[23]
     mov ecx,5
     repe cmpsb
     jnz else5
```

```
                        call crlf
        mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
        inc score
        mov arr_L2[4],0
        jmp next

      else5:
        call crlf
          mWrite<" Your entered word not found!!!!",0dh,0ah>
        dec lives
      next:
          mov eax,1500
          call delay
          call clrscr
          mov al,score
          cmp al,10
          jl outter
              JZ quit

nolife:
     call crlf
 call crlf
 call crlf
 call crlf
 call crlf
     mWrite<"      NO LIFE LEFT TO PLAY",0>
     mWrite<"       <------ * GAME OVER!!!!! * ------>",0>
 ret

quit:
     call crlf
     ret
Level2 endp
;-----------------------------------------------

;------------------*Level3*------------------
Level3 Proc

   call crlf
   call crlf
   call crlf
   mov ecx,1000
   call delay
   mWrite<" --->>> LEVEL 2 SUCCESSFULLY COMPLETED, NOW PASS NEXT LEVEL <<<---",0
outter:
       call crlf
   call crlf
   call crlf
   mWrite<"       > LEVEL 3: <",0>
```

```
call crlf
call crlf
   cmp lives,0
je nolife

   mWrite<" >> Lives : ",0>
   movzx eax,lives
   call writedec
   mWrite<"     >> Score : ",0>
   movzx eax,score
   call WriteDec
call crlf
call crlf
mov edx,offset file_L3
call read_file
call crlf

mov edx,offset str1
call writestring

mov  edx,OFFSET input
   mov  ecx,9
   call ReadString


mov al,arr_L3[0]
cmp al,1
jne else1
cld
mov esi,offset input
mov edi,offset word_list2[0]
mov ecx,7
repe cmpsb
jnZ else1
call crlf
mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
inc score
mov arr_L3[0],0
jmp next

else1:
     mov al,arr_L3[1]
     cmp al,1
     jne else2
     cld
     mov esi,offset input
     mov edi,offset word_list2[7]
     mov ecx,5
     repe cmpsb
   jnz else2
            call crlf
   mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
   inc score
   mov arr_L3[1],0
   jmp next

else2:
     mov al,arr_L3[2]
   cmp al,1
   jne else3
   cld
   mov esi,offset input
   mov edi,offset word_list2[12]
   mov ecx,6
   repe cmpsb
   jnz else3
            call crlf
   mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
   inc score
   mov arr_L3[2],0
   jmp next

  else3:
     mov al,arr_L3[3]
   cmp al,1
   jne else4
   cld
   mov esi,offset input
   mov edi,offset word_list2[18]
   mov ecx,6
   repe cmpsb
   jnz else4
            call crlf
   mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
   inc score
   mov arr_L3[3],0
   jmp next

  else4:
     mov al,arr_L3[4]
   cmp al,1
   jne else5
   cld
   mov esi,offset input
   mov edi,offset word_list2[24]
   mov ecx,4
   repe cmpsb
   jnz else5
```

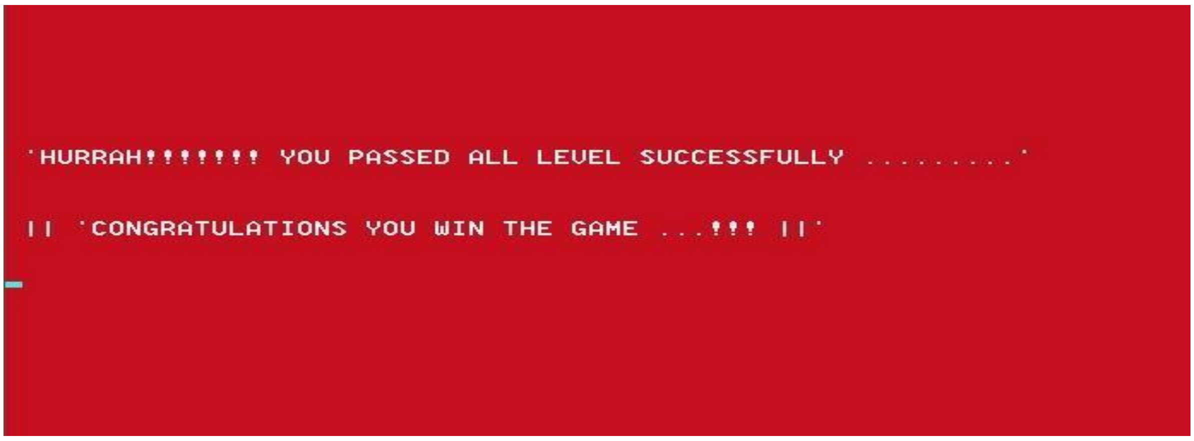
```
                     call crlf
         mWrite <" Your entered word Successfully found!!!!",0dh,0ah>
         inc score
         mov arr_L3[4],0
         jmp next

        else5:
         call crlf
             mWrite<" Your entered word not found!!!!",0dh,0ah>
         dec lives

    next:
            mov eax,1500
            call delay
            call clrscr
            mov al,score
            cmp al,15
            jl outter
            je success

        success:
                call crlf
                call crlf
                call crlf
                mov ecx,2500
                call delay
            call crlf
            call crlf
            call crlf
                mWrite<" 'HURRAH!!!!!!! YOU PASSED ALL LEVEL SUCCESSFULLY .........'",0>
                call crlf
                            call crlf
                            call crlf
                mWrite<" || 'CONGRATULATIONS YOU WIN THE GAME ...!!! ||'",0>
                            jmp quit
nolife:
     call crlf
 call crlf
 call crlf
 call crlf
 call crlf
     mWrite<" NO LIFE LEFT TO PLAY",0>
     mWrite<"       <------ * GAME OVER!!!!! * ------>",0>
 ret

quit:
 call Crlf
 call Crlf
     ret
```

```
Level3 endp
;-----------------------------------------------

;------------------Read_File***------------------
Read_File proc

call OpenInputFile
mov fileHandle,eax
cmp eax,INVALID_HANDLE_VALUE
jne file_ok
mWrite <" ERROR: Cannot open file...!!!",0dh,0ah>
jmp quit

file_ok:
        mov edx,OFFSET buffer
        mov ecx,BUFFER_SIZE
        call ReadFromFile
        jnc check_buffer_size
        mWrite " ERROR: In Reading file...!!!"
        call WriteWindowsMsg
        jmp close_file

check_buffer_size:
        cmp eax,BUFFER_SIZE
            jb buf_size_ok
        mWrite <"ERROR: Buffer too small for the file...!!!",0dh,0ah>
        jmp quit

buf_size_ok:
            mov buffer[eax],0
            mov edx,OFFSET buffer
                        call WriteString
            call Crlf
close_file:
           mov eax,fileHandle
           call CloseFile

quit:
     ret
Read_File endp
;-----------------------------------------------

;------------------*Instruction*------------------
Instruction PROC

mov edx,offset file_L4
call Read_File
call crlf
ret
```

```
Instruction endp
;---------------------------------------------------

;------------------*Write_File*-----------------
Write_File PROC

mov edx,OFFSET filename
call CreateOutputFile
mov fileHandle,eax
cmp eax, INVALID_HANDLE_VALUE
jne file_ok
mWrite<" ERROR: Cannot Create file...!!!",0dh,0ah,0>
jmp quit

file_ok:
   mov eax,0
   cld
   mov al,score
   mov edi,offset char
   stosd
      mov eax,fileHandle
      mov edx,offset char
      mov ecx,4
      call WriteToFile
      call CloseFile

quit:
    ret
Write_File endp
;---------------------------------------------------

end main
```

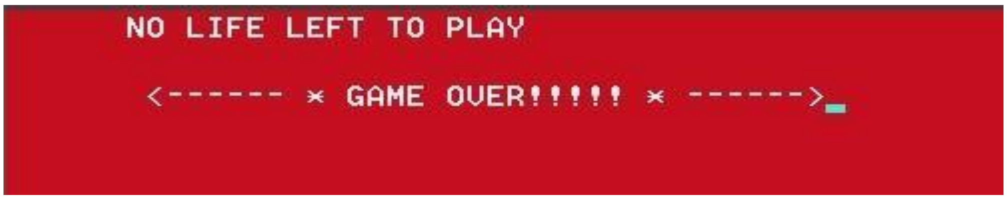
<u>RESULT SOFTWARE SIMULATION AND DISCUSSION:</u>

Our Crossword Puzzle Game Program perfectly works and pass all possible test cases. We also debug our whole code using Visual Studio 2019 IDE.
Some of the test cases are given below:



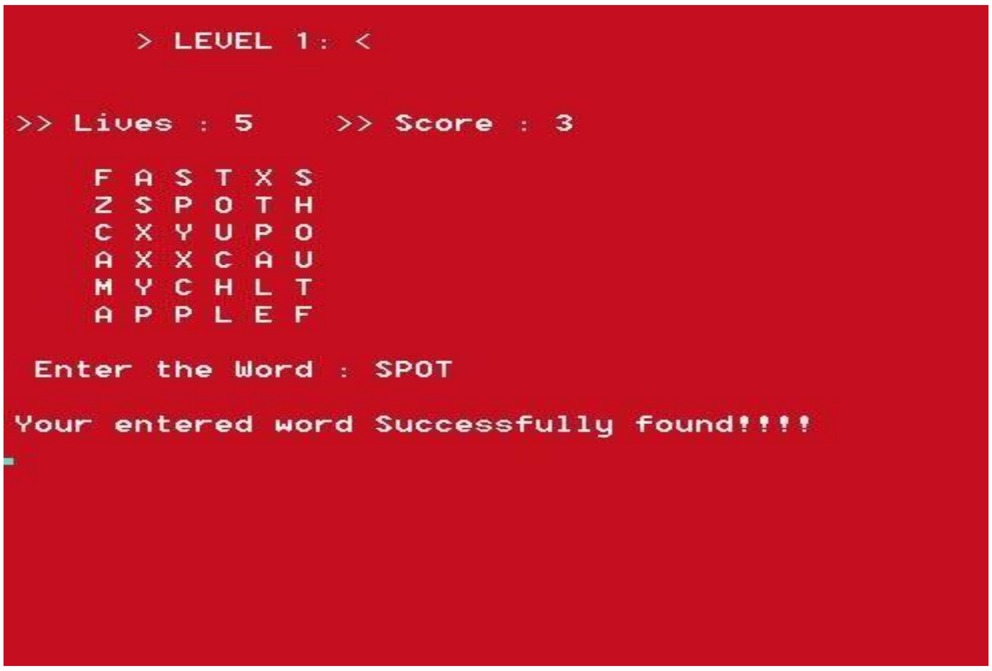
```
'HURRAH!!!!!!!! YOU PASSED ALL LEVEL SUCCESSFULLY .........'

|| 'CONGRATULATIONS YOU WIN THE GAME ...!!! ||'
```

This is the first test case in which the lives are not ended means if no life is wasted and you clear all the three levels then the program prints these lines. In the other case if you clear all the three levels and some of your life are wasted but they are not ended so the program also prints the same lines.



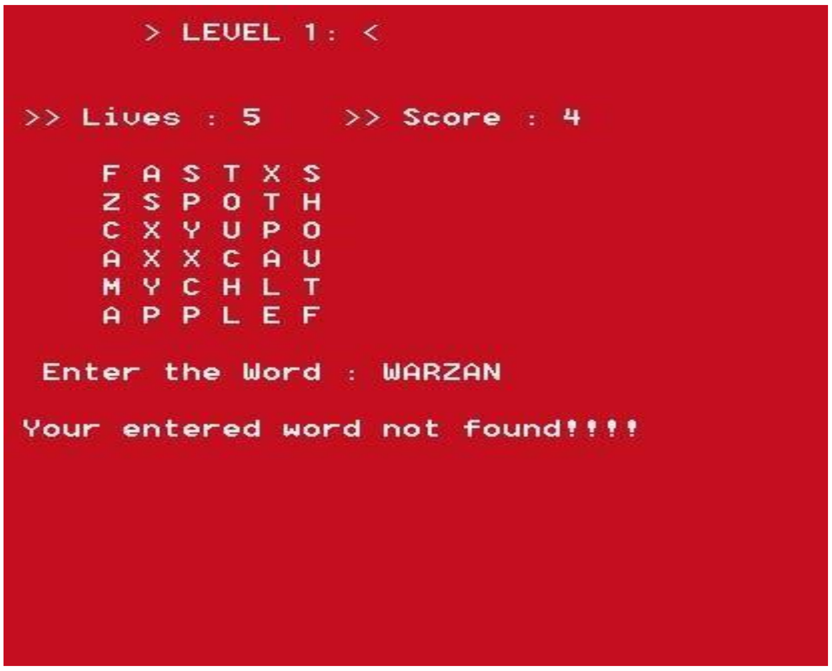
```
      NO LIFE LEFT TO PLAY

       <------ * GAME OVER!!!!! * ------>
```

This is the case in which your lives are ended so the program prints this on screen. It means that even your lives are ended in the 1st level or in the 2nd level or in the 3rd level no matter, whenever your lives are wasted means you have zero lives then the program prints these lines.



```
       > LEVEL 1: <

>> Lives : 5     >> Score : 3

    F A S T X S
    Z S P O T H
    C X Y U P O
    A X X C A U
    M Y C H L T
    A P P L E F

 Enter the Word : SPOT

Your entered word Successfully found!!!!
```

This is the test case in which the user entered the correct or the right letter of word such that in the implementation programming code it successfully compares the two strings then it prints these lines.



```
       > LEVEL 1: <

>> Lives : 5     >> Score : 4

    F A S T X S
    Z S P O T H
    C X Y U P O
    A X X C A U
    M Y C H L T
    A P P L E F

 Enter the Word : WARZAN

Your entered word not found!!!!
```

This is the test case in which the user entered the incorrect or the wrong letter of word such that in the implementation programming code it does not compares the two strings then it prints these lines.

CONCLUSION:

This Crossword Game has many benefits. It is many beneficial in the future life for students as well as for the people for all ages. This game helps us to improves mind ability as well as improves our focus to solve problems. To solve problem different strategies implemented in this way its help to improve strategy.

REFERENCES:

We take an idea from Wikipedia and some other resources websites.

"Word Search Strategies to Help You Solve Any Word Search Puzzle".
"https://en.wikipedia.org/wiki/Word_search"